2025. 11.
졸업 작품 프로젝트 발표

DCT 비가시 워터마킹 이미지 기반 창작물 플랫폼

Team.DDW

팀장 : 정민재
팀원 : 유혁준, 임건희

CONTENTS

01 프로젝트 개요 - 연구 배경 및 필요성

- 디지털 창작물의 무단 복제와 저작권 침해의 심각성

- 원본 출처의 불분명함

- 가시적 워터마크의 한계

01 프로젝트 개요 - 가시적 워터마크

- 콘텐츠 자체를 시각적으로 방해
- 쉽게 훼손, 제거될 수 있음

01 프로젝트 개요 - 연구 목적 및 주제 선정

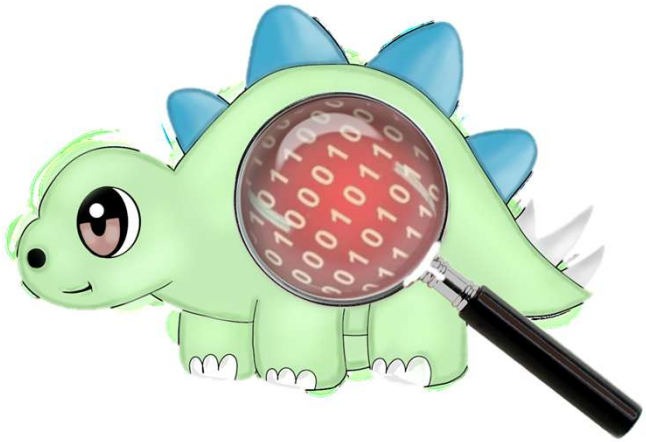
- 이미지 창작물 저작권 분쟁 및 무단 유포 사례 증가 (예: 일러스트, 사진, 웹툰 등)

- DCT(이산 코사인 변환) 워터마킹 기법을 통한 보이지 않는 워터마크

- 비가시적인 워터마크를 활용한 플랫폼 개발

02 관련 이론 및 기술적 배경

- 디지털 워터마킹 개념



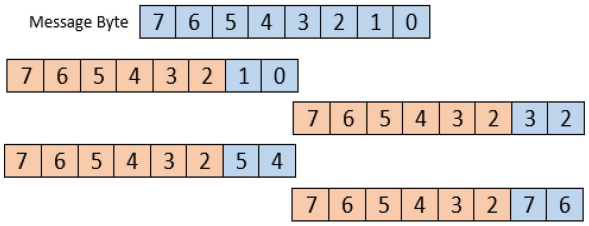
- 스테가노그래피

디지털 이미지, 영상, 오디오 등 디지털 콘텐츠에 식별 정보나 저작권 정보를 삽입하여

불법 복제나 무단 배포를 방지하고 출처를 밝힐 수 있도록 하는 기술

02 관련 이론 및 기술적 배경

- 공간영역 워터마킹과 주파수 영역 워터마킹



- 공간 영역 : 픽셀 값 직접 수정 (LSB 등), 변형 및 변조에 취약
- 주파수 영역 : DCT, DFT, DWT 등 변환 후 계수에 삽입, 압축/노이즈에 강함

02 관련 이론 및 기술적 배경 - DCT 기반 워터마킹 원리

중간 주파수 대역 계수 조작 기법

DCT 변환 후 8x8 블록의 중간 주파수 계수(예: (4,4))에 워터마크 비트를 삽입

저주파는 시각적으로 민감, 고주파는 압축에 약하므로 중간 주파수 영역이 적합

02 DCT 기반 워터마킹 원리 - 워터마크 삽입 및 복원 알고리즘 구조

삽입: 이미지 → 8x8 블록 분할 → DCT → 계수 조작 → 역DCT → 저장

복원: 이미지 → 8x8 블록 분할 → DCT → 계수 판독 → 메시지 복원

02 관련 기술 스택

- 프론트엔드 : Next.js

사용자 인터페이스, 이미지 업로드 및 결과 표시

02 관련 기술 스택

- 백엔드 : Flask, OpenCV, Python

API 서버, 이미지 처리, 스타일링

02 관련 기술 스택

- 워터마크 엔진 : OpenCV, NumPy,Pillow

DCT 워터마킹 알고리즘 구현 및 이미지 입출력

02 관련 기술 스택

- 배포

Next.js 기반 프로젝트에 최적화된 클라우드 배포

03 시스템 설계 및 구현 - 서비스 개념

기본적인 이미지 창작물 플랫폼 형태 구현

사용자가 이미지를 업로드하면, 사용자를 특정하는 DCT 기반 워터마크를 삽입하여
저작권 침해 방지, 출처 인증

03 시스템 설계 및 구현 - 사용자 흐름

회원가입/로그인 → 이미지 업로드 → 워터마크 삽입 → 결과 다운로드 및 추출

사용자 인증 및 관리

PNG/JPEG 등 다양한

이미지 포맷 지원

입력 메시지를 DCT 계수에 은닉하여

이미지 품질 저하 없이 워터마크 삽입

업로드된 이미지에서 워터마크

메시지 추출을 통해 출처 확인

04 핵심 구현 코드 - 워터마크 삽입

```
def encrypt(self, image_path: str, message: str, output_path: str, output_format: str) -> None:
    img = self._prepare_image(image_path)
    height, width, channels = img.shape

    # 메시지를 비트 시퀀스로 변환
    bits = self._string_to_bits(message)

    # 삽입할 수 있는 최대 비트 수 계산
    blocks_height = height // self.BLOCK_SIZE
    blocks_width = width // self.BLOCK_SIZE
    max_bits = blocks_height * blocks_width * channels

    if len(bits) > max_bits:
        raise ValueError(f"텍스트 최대 길이 초과. 최대 비트: {max_bits}, 필요: {len(bits)}")

    bit_index = 0
    modified_img = np.copy(img)

    for channel in range(3):  # RGB 채널
        if bit_index >= len(bits):
            break
        channel_data = modified_img[:, :, channel]

        for y in range(0, blocks_height * self.BLOCK_SIZE, self.BLOCK_SIZE):
            for x in range(0, blocks_width * self.BLOCK_SIZE, self.BLOCK_SIZE):
                if bit_index >= len(bits):
                    break
                block = channel_data[y:y+self.BLOCK_SIZE, x:x+self.BLOCK_SIZE]

                # 비트 삽입
                modified_block = self._embed_bit_in_dct_block(block, bits[bit_index])
                modified_img[y:y+self.BLOCK_SIZE, x:x+self.BLOCK_SIZE, channel] = modified_block

                bit_index += 1

    # 이미지 저장
    self._save_image(modified_img, output_path, output_format)
def _embed_bit_in_dct_block(self, block: np.ndarray, bit: int) -> np.ndarray:
    block_float = np.float32(block)
    dct_block = cv2.dct(block_float)

    # 중간 주파수 계수 (4,4)에 비트 삽입
    if bit == 1:
        if dct_block[4, 4] < self.THRESHOLD:
            dct_block[4, 4] = self.THRESHOLD + self.QUANTIZATION_FACTOR
    else:
        if dct_block[4, 4] > -self.THRESHOLD:
            dct_block[4, 4] = -self.THRESHOLD - self.QUANTIZATION_FACTOR

    idct_block = cv2.idct(dct_block)
    return np.clip(idct_block, 0, 255).astype(np.uint8)
```

이미지를 8×8 블록 단위로 DCT 변환한 후, 중간 주파수 계수에 사용자 정보를 삽입

중간 주파수 계수 조작을 통해 워터마크 삽입,

품질 저하를 최소화하여 삽입 정보를 은닉

04 핵심 구현 코드 - 워터마크 추출

```python
def decrypt(self, image_path: str) -> str:
    img = self._prepare_image(image_path)
    height, width, channels = img.shape

    blocks_height = height // self.BLOCK_SIZE
    blocks_width = width // self.BLOCK_SIZE

    extracted_bits = []
    consecutive_ones = 0

    for channel in range(3):  # RGB 채널
        channel_data = img[:, :, channel]

        for y in range(0, blocks_height * self.BLOCK_SIZE, self.BLOCK_SIZE):
            for x in range(0, blocks_width * self.BLOCK_SIZE, self.BLOCK_SIZE):
                if consecutive_ones >= 16:  # 종료 시퀀스 확인
                    break

                block = channel_data[y:y+self.BLOCK_SIZE, x:x+self.BLOCK_SIZE]
                bit = self._extract_bit_from_dct_block(block)
                extracted_bits.append(bit)

                if bit == 1:
                    consecutive_ones += 1
                else:
                    consecutive_ones = 0

            if consecutive_ones >= 16:
                break
        if consecutive_ones >= 16:
            break

    if consecutive_ones >= 16:
        extracted_bits = extracted_bits[:-consecutive_ones]

    return self._bits_to_string(extracted_bits)
def _extract_bit_from_dct_block(self, block: np.ndarray) -> int:
    block_float = np.float32(block)
    dct_block = cv2.dct(block_float)
    return 1 if dct_block[4, 4] > 0 else 0
```

워터마크가 삽입된 이미지를 다시 DCT 변환하여 중간 주파수 계수에서 숨겨진 비트를 읽어 복호화

삽입된 워터마크 정보를 추출하여 삽입된 이미지의 출처와 업로더 정보를 확인

05 시연 영상

유튜브 링크

06 결론 및 향후 개선 방향

DCT 기반 워터마킹 기술로 이미지 품질 저하 없이

원본 출처를 확인 가능, 무단 도용 방지

향후 DCT 워터마킹을 프레임 단위로 적용하여 동영상 전체에

삽입함으로써 초기 목적이었던 동영상 콘텐츠에서도 같은 기능을 구현

THANK YOU